Sensor Data Reconstruction and Anomaly Detection using Bidirectional Recurrent Neural Network

Seongwoon Jeong*, Max Ferguson, Kincho H. Law

Department of Civil and Environmental Engineering, Stanford University
Stanford, CA 94305, USA

ABSTRACT

With advances in sensing and communication technologies, engineering systems are now commonly instrumented with sensors for system monitoring and management. Occasionally, when sensors become malfunction, it is advantageous to automatically determine faulty sensors in the system and, if possible, recover missing or faulty data. This paper investigates the use of machine learning techniques for sensor data reconstruction and anomaly detection. Specifically, bidirectional recurrent neural network (BRNN) is employed to build a data-driven model for sensor data reconstruction based on the spatiotemporal correlation among the sensor data. The reconstructed sensor data can be used not only for recovering the data of the faulty sensors, but also for detecting anomalies based on an analytical redundancy approach. The proposed method is tested with vibration data based on a numerical simulation of a sensor network for bridge monitoring application. In terms of prediction accuracy, the results show that the BRNN-based sensor data reconstruction method performs better than other existing sensor data reconstruction methods. Furthermore, the sensor data reconstructed can be used to detect and isolate the anomalies caused by faulty sensors.

Keywords: Sensor data reconstruction, anomaly detection, sensor validation, bidirectional recurrent neural network

1. INTRODUCTION

In today's engineering applications, sensors serve a critical role in monitoring and controlling the operations of an engineering system. Data collected from sensors enables fault detection and evaluation of the health of the system [1, 2]. However, sensors can become faulty due to various reasons, including noise, poor installation, harsh environment, aging, battery problem, etc. [3]. Faulty sensors generate various types of faulty data, such as bias, drift, complete failure and precision degradation [4]. Some faults (e.g., complete failure of a sensor) are relatively easy to detect, whereas other faults (e.g., precision degradation in the sensor output) are relatively difficult to identify. Faulty sensors not only cause permanent loss of valuable data, but also can lead to inaccurate diagnosis of the monitored system. For successful deployment of a sensing system, sensors need to be validated and the faulty data from faulty sensors should be reconstructed.

Sensor validation consists of three basic problems, namely sensor fault detection, fault isolation and fault reconstruction [4]. The three faulty sensor problems are closely related to each other. For example, sensor data reconstruction (i.e., fault reconstruction) can be used to recover the data of faulty sensors detected by fault detection and isolation. Furthermore, sensor data reconstruction can be used to detect and isolate faulty sensors via an analytical redundancy method that captures anomalies based on the discrepancy between the measurement data and the reconstructed data [5]. Abilities to accurately reconstruct sensor data are important for maintaining a healthy sensor network. This paper discusses a highly accurate sensor data reconstruction method that can potentially be used for data recovery, as well as for anomaly detection and isolation.

Typically, sensor data reconstruction methods attempt to predict the data of a target sensor based on the data collected by other sensors. One approach is to use the spatial correlation among the sensors. Sensor data reconstruction methods based on spatial correlation have been proposed using statistical and machine learning approaches, such as principal component analysis (PCA) [6], minimum mean square error (MMSE) estimation [7], support vector regression (SVR) [8] and artificial neural network (ANN) [9, 10, 11, 12]. While the studies have shown that the spatial correlation-based methods can

[*] e-mail: swjeong3@stanford.edu

reconstruct the sensor data well, the accuracy can be improved by considering additional information, such as the temporal correlation among the sensor data. For example, Kullaa [7] extends the MMSE-based sensor validation approach by utilizing the linear spatiotemporal correlation among the sensors. To handle the nonlinear spatiotemporal correlation among the sensor data, various architectures of ANN have been investigated. Moustapha and Selmic [13] propose a sensor validation method based on the recurrent neural network (RNN) to take into consideration both the spatial and the temporal correlations among the data. So far, most studies have considered only the data from the past to reconstruct the sensor data. A future context, if available, can help improve the accuracy of data reconstruction and anomaly detection.

This paper describes a data-driven sensor data reconstruction and anomaly detection method that leverages the spatial and the past and future temporal correlations among the sensor data. Specifically, bidirectional recurrent neural network (BRNN) [14] is employed to construct a sensor data reconstruction model. In the next section, BRNN-based method for sensor data reconstruction will be described. We then discuss faulty sensor detection and isolation using the data reconstructed by the BRNN-based method. The proposed methods are demonstrated and validated with bridge monitoring application using numerical simulations. The paper concludes with a brief summary and discussion.

2. BRNN-BASED SENSOR DATA RECONSTRUCTION METHOD

This section describes the BRNN-based method for sensor data reconstruction. First, the architecture of BRNN model for sensor data reconstruction is described. The model training and data reconstruction processes are then discussed.

2.1 BRNN model for sensor data reconstruction

Consider a sensor network consisting of *N* input sensors and a single output (target) sensor, each with time series measurement data denoted, respectively, as $\boldsymbol{x}(t) = \{x_1(t), \ldots, x_i(t), \ldots, x_N(t)\}$ and $y(t)$ where $t = 1, \ldots, T$, the number (or length) of measurements (or time steps) per each sensor, and $i = 1, \ldots, N$, the number of input sensors. The sensor data reconstruction problem can be treated as a supervised learning or regression problem for estimating the predicted outputs $\hat{y}(t)$, where $t = 1, \ldots, T$, using the input measurement data $\boldsymbol{x}(t)$, such that the predictions $\hat{y}(t)$ are good estimations of the actual output measurements $y(t)$.

Figure 1 depicts the neural network architecture of the BRNN model and the computational flows between the layers of the neural network per each time step and among the different time steps. For each time step t, the neural network consists of three layers:

- An input layer consists of *N* input units containing the data $\boldsymbol{x}(t)$ from the *N* input sensors.
- An output layer consists of a single output unit containing the predicted value $\hat{y}(t)$ of the target sensor.
- A hidden layer consists of *M* forward hidden units and *M* backward hidden units containing the values denoted by $\boldsymbol{h}^f(t)$ and $\boldsymbol{h}^b(t)$, respectively.

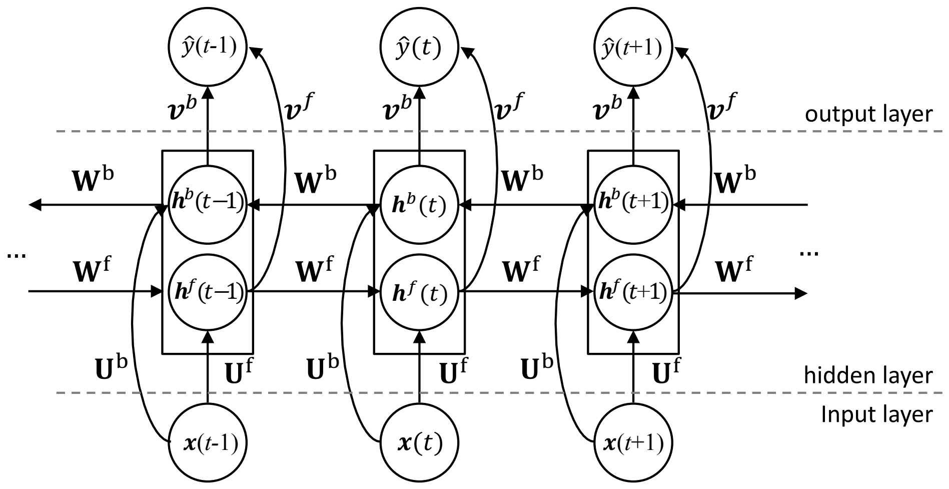


Figure 1. Structure of bidirectional recurrent neural network with a single hidden layer

As shown in Figure 1, the hidden units at each time step t are connected with the adjacent hidden units at time steps $t-1$ and $t+1$. Specifically, the forward hidden units are fully connected with the data flow from $\boldsymbol{h}^f(t-1)$ to $\boldsymbol{h}^f(t)$ and the backward hidden units with data flow from $\boldsymbol{h}^b(t+1)$ to $\boldsymbol{h}^b(t)$. As depicted using the directional arrows in Figure 1, the values of the hidden units are updated at each iteration as:

$$\boldsymbol{h}^f(t) = f\left(\mathbf{U}^{\mathrm{f}}\boldsymbol{x}(t) + \mathbf{W}^{\mathrm{f}}\boldsymbol{h}^f(t-1) + \boldsymbol{b}^f\right) \tag{1}$$

$$\boldsymbol{h}^b(t) = f\left(\mathbf{U}^{\mathrm{b}}\boldsymbol{x}(t) + \mathbf{W}^{\mathrm{b}}\boldsymbol{h}^b(t+1) + \boldsymbol{b}^b\right) \tag{2}$$

where (1) $\mathbf{U}^{\mathrm{f}}$ and $\mathbf{U}^{\mathrm{b}}$ denote the $M \times N$ weights connecting the N input units to the M forward hidden units and to the M backward hidden units, respectively; (2) $\mathbf{W}^{\mathrm{f}}$ and $\mathbf{W}^{\mathrm{b}}$ denote the $M \times M$ weights between the hidden units of two successive time steps; and (3) $\boldsymbol{b}^f$ and $\boldsymbol{b}^b$ denote the $M \times 1$ biases for the hidden units. In addition, $f(\cdot)$ denotes an activation function which can be a linear or nonlinear function [15]. In this study, the hyperbolic tangent function is selected as the activation function for both the forward and the backward hidden units:

$$f(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})} \tag{3}$$

Once $\boldsymbol{h}^f(t)$ and $\boldsymbol{h}^b(t)$ of the hidden units are obtained, the predicted output data $\hat{y}(t)$ at the time step t can be computed as:

$$\hat{y}(t) = \boldsymbol{v}^f \cdot \boldsymbol{h}^f(t) + \boldsymbol{v}^b \cdot \boldsymbol{h}^b(t) + c \tag{4}$$

where $\boldsymbol{v}^f$ and $\boldsymbol{v}^b$ denote a vector of M weighting values that connect the output unit with the forward and the backward hidden units, respectively, and c denotes the bias for the output unit. From Eqs. (1), (2) and (4), it can be seen that the reconstructed output data $\hat{y}(t)$ at time step t involves the past context with the forward hidden units with $\boldsymbol{h}^f(t-1)$, the present context from the current input data $\boldsymbol{x}(t)$ and the future context with the backward hidden units with $\boldsymbol{h}^b(t+1)$.

2.2 Training the BRNN model

The BRNN model is trained with a dataset $\{(\boldsymbol{x}(t), y(t)),\ t = 1, 2, \dots, T\}$ of measurements from the input and output sensors. First, for each sensor, the time series data is normalized to have a zero mean and scaled to have values within the range from -1 to 1 for good convergence during the training process [16]. The normalized and scaled data is then used to train the BRNN model by calibrating the parameters (i.e., the weights $\mathbf{U}^{\mathrm{f}}, \mathbf{U}^{\mathrm{b}}, \mathbf{W}^{\mathrm{f}}, \mathbf{W}^{\mathrm{b}}, \boldsymbol{v}^f, \boldsymbol{v}^b$ and the biases $\boldsymbol{b}^f, \boldsymbol{b}^b, c$) to minimize the difference between the actual measurement $y(t)$ and the predicted output $\hat{y}(t)$ for each time step $t = 1, \dots, T$. The backpropagation through time (BPTT) algorithm is employed for training the BRNN model [17]. The BPTT algorithm consists of three basic steps [14]:

- In the *forward pass*, the predicted output data $\hat{y}(t)$ is calculated for $t = 1, \dots, T$ using the input data $\boldsymbol{x}(t)$ through Eqs. (1), (2) and (4).
- In the *backward pass*, the gradients of the loss (i.e., the difference) between the measured output data $y(t)$ and the predicted output data $\hat{y}(t)$, $t = 1, \dots, T$, with respect to the parameters are computed. The gradients are then propagated through the connections between the layers and among the different time steps.
- For *parameter update*, the parameters of the BRNN model are adjusted based on the propagated gradients via an optimization procedure, such as the Adaptive Moment Estimation (Adam) algorithm [18].

These three steps are repeated until the change of the loss is within a predefined threshold or the number of epochs (i.e., the number of times that the entire training dataset is being processed) reaches a prescribed limit.

2.3 Data reconstruction using trained BRNN model

Once the BRNN model is trained, the time series output data $\hat{y}'(t)$ of length T' of the target sensor can be estimated using the corresponding time series data $\boldsymbol{x}'(t)$ from the input sensors. The sensor data reconstruction consists of two steps. First, the input data $\boldsymbol{x}'(t)$, $t = 1, \dots, T'$, are normalized and scaled using the normalization and scaling factors obtained during the training process. Second, the BRNN model is then used to reconstruct the output data $y'(t)$ using Eqs. (1), (2) and (4).

The reconstructed output data can have at least three different usages. First, when both the input data $\boldsymbol{x}'(t)$ and the output data $y'(t)$ are available, the BRNN model can be evaluated by computing the root-mean-square-error (RMSE) as:

$$\epsilon = \sqrt{\frac{\sum_{t=1}^{T'}(\hat{y}'(t) - y'(t))^2}{T'}} \tag{5}$$

The reconstruction error ϵ can be used as an indicator to detect potential anomaly of the output sensor. Second, if the target sensor is known to be faulty, the faulty measurement data $y'(t)$ can be replaced by the reconstructed data $\hat{y}'(t)$. Third, as to be discussed in the next section, if the time series data of a sensor indicates the possibility of anomalies, the reconstruction error ϵ computed by Eq. (5) can be used to detect and isolate the anomalies.

3. ANOMALY DETECTION USING RECONSTRUCTED SENSOR DATA

Among the widely used fault detection and isolation (FDI) methods is the analytical redundancy approach that determines anomalies by comparing the original measurement data with the reconstructed data [5, 19, 20]. This section describes the sensor validation process, which includes anomaly detection and isolation, using the analytical redundancy approach based on the reconstructed sensor data.

3.1 Anomaly detection

The goal of anomaly detection is to determine the existence of one or more faulty sensors in a sensor network. The basic idea of the anomaly detection is that the existence of a faulty sensor will lead to some "measurable" difference between the measured data and the reconstructed data.

For the anomaly detection of a system with N sensors, it is necessary to consider the N combinations of the input and output pairs of sensors. That is, for the n^{th} combination, the n^{th} sensor is considered as the output (target) sensor while all other sensors are treated as input sensors. Therefore, prior to the anomaly detection, the BRNN models $g_1(\cdot), \ldots, g_N(\cdot)$ for all N input-output combinations are first created and trained. Furthermore, the trained models are employed for testing on a number of trial datasets to establish the confidence levels for the anomaly detection. In this study, the confidence thresholds $C_1, \ldots, C_N$ are defined for each input-output combination based on a 95% confidence interval established for the trained BRNN models $g_1(\cdot), \ldots, g_N(\cdot)$. As summarized in Algorithm 1, the anomaly detection proceeds as follows:

(1) Select an output, say n^{th}, sensor from the N sensors in the network.
(2) Reconstruct the output data for the n^{th} sensor using the corresponding BRNN model $g_n(\cdot)$ and the input data from all other sensors through Eqs. (1), (2) and (4) (see line 3 of Algorithm 1).
(3) Compute the reconstruction error of the target sensor using Eq. (5) (see line 4 of Algorithm 1).
(4) Repeat steps (1), (2) and (3) for all N input-output combinations (see line 1 of Algorithm 1).
(5) If the reconstruction error from any of the N input-output combinations exceeds the defined confidence thresholds, the system is assumed to have at least one faulty sensor (see line 6 of Algorithm 1).

Algorithm 1. Anomaly detection using BRNN model

```
Input:
    $g_1(\cdot), \ldots, g_N(\cdot)$: trained BRNN models
    $C_1, \ldots, C_N$: thresholds for anomaly detection
    $\mathbf{D}' = [\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_i, \ldots, \boldsymbol{d}'_N]$: preprocessed testing dataset collected by the $N$ sensors where $\boldsymbol{d}'_i$ consists of the
    measurement data $\{x'_i(1), \ldots, x'_i(T')\}$ for sensor i.
1:  for each $n \in [1, N]$ do
2:      Set output $\boldsymbol{y}' \leftarrow \boldsymbol{d}'_n$ and input $\mathbf{D}' \backslash \boldsymbol{d}'_n = [\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_{n-1}, \boldsymbol{d}'_{n+1}, \ldots, \boldsymbol{d}'_N]$
3:      Compute $\hat{\boldsymbol{y}}'$ of the $n^{th}$ sensor using BRNN model $g_n(\cdot)$ and input $\mathbf{D}' \backslash \boldsymbol{d}'_n$ according to Eqs. (1), (2) and (4)
4:      Compute the reconstruction error $\epsilon_n$ for the $n^{th}$ sensor between $\boldsymbol{y}'$ and $\hat{\boldsymbol{y}}'$ using Eq. (5)
5:  end for
6:  if $\exists n \in [1, N]$ $\epsilon_n > C_n$, then there exist anomalies (i.e., at least one faulty sensor exists among the $N$ sensors)
```

3.2 Anomaly isolation

The goal of anomaly isolation is to identify the faulty sensor in a system. A simple approach for anomaly isolation would be to infer the sensors with reconstruction errors exceeding their thresholds as faulty sensors. However, this approach is prone to false positive error because a faulty sensor can increase the reconstruction error not only when the faulty sensor is the output sensor, but also when the faulty sensor is one of the input sensors. Another approach is to consider the sensor with the largest reconstruction error in comparison to the threshold as the faulty sensor. However, this approach is prone to false negative error when there exist multiple faulty sensors. To avoid the likelihood of false positive and false negative errors, an elimination approach is employed in this work.

The basic idea is that the faulty sensor causes only local effects and, thus, the anomaly disappears if all faulty sensors are removed from the sensor network [21]. The procedure follows the anomaly detection algorithm by iteratively eliminating the faulty sensors and their measurement data. Using the test dataset for anomaly detection, a set of potentially faulty sensors, where the reconstruction errors exceed the confidence thresholds, are first identified. The elimination procedure for isolating the faulty sensors then proceeds as shown in algorithm 2 as follows:

(1) Assign the sensor with the highest reconstruction error as a faulty sensor and remove the sensor from the network.
(2) Train and construct the BRNN models and compute the confidence threshold with the rest of the sensors in the network without the previously identified faulty sensors.
(3) Perform the anomaly detection procedure (Algorithm 1) to identify faulty sensor, if any.

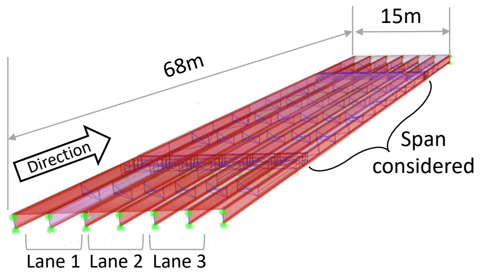
The procedure is repeated until no further faulty sensors are identified.

Algorithm 2. Anomaly isolation based on elimination approach

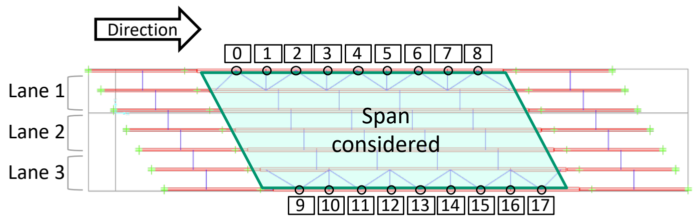
```
Input:
    $X = \{1, \ldots, N\}$: a list of all $N$ sensors
    $S = \{n \in X | \epsilon_n > C_n\}$: a list of potentially faulty sensors with $\epsilon_n > C_n$
    $F = \emptyset$: a list of faulty sensors, initially set to null
    $\mathbf{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N]$: preprocessed training dataset collected by the $N$ sensors
    $\mathbf{D}' = [\boldsymbol{d}'_1, \ldots, \boldsymbol{d}'_N]$: preprocessed testing dataset collected by the $N$ sensors
1:  Loop until $S = \emptyset$
2:      Append a sensor $i$ with the highest reconstruction error in the list $S$ to the list $F$
3:      Update the training dataset $\mathbf{D} \leftarrow \mathbf{D} \backslash \boldsymbol{d}_i$ and the testing dataset $\mathbf{D}' \leftarrow \mathbf{D}' \backslash \boldsymbol{d}'_i$
4:      Construct and train BRNN models $g_n(\cdot)$, $n \in X \backslash F$ with the remaining training dataset $\mathbf{D}$ from sensors
        $X \backslash F$
5:      Compute the confidence threshold $C_n$, $n \in X \backslash F$
6:      Perform Algorithm 1 for anomaly detection with the testing dataset $\mathbf{D}'$
7:      Update the list of the potentially faulty sensors $S \leftarrow \{n \in X \backslash F | \epsilon_n > C_n\}$
8:  end loop
9:  Identify the sensors in the list $F$ as potentially faulty
```

4. NUMERICAL SIMULATIONS

Numerical simulations are conducted to demonstrate and evaluate the BRNN-based sensor data reconstruction method. As shown in Figure 2(a), a finite element (FE) model for the Telegraph Road Bridge located in Monroe, Michigan is employed [22, 23]. Here, the FE model is constructed using CSI Bridge [24], a commonly used bridge analysis tool. Vertical vibration responses due to randomly traveling vehicles are simulated and recorded at 18 (sensor) locations on the two exterior girders, as shown in Figure 2(b).



(a) 3-D FE model



(b) Sensor layout

Figure 2. Finite element model and sensor layout of the Telegraph Road Bridge

To emulate the load conditions on the bridge, moving vehicles are randomly defined using the variables summarized in Table 1. The vehicle types (labelled as auto, H-20, HS-20 and HS-25), as shown in Figure 3, and their loads are defined based on the AASHTO (American Association of State Highway and Transportation Officials) standard [25]. Dynamic time history analyses are performed with randomized traffic patterns on the bridge to generate the vertical vibration response (sensor) data. Each analysis assumes a 10-second duration with incremental time step size of 0.005 (i.e., a sampling rate of 200 Hz). To generate sufficient amount of data for the data-driven method, a total of 300 analyses are conducted with randomized moving vehicle loads. Through 300 analyses, 600,000 data points per sensor location are collected and preprocessed (i.e., normalized and scaled) as shown in Figure 4.

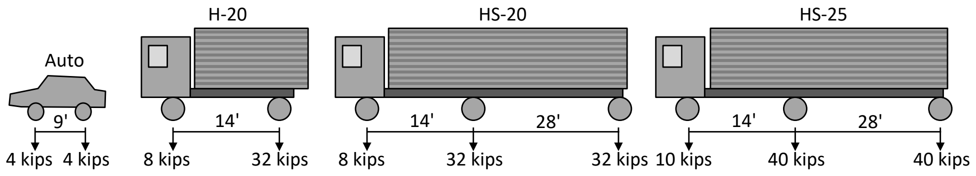


Figure 3. Vehicle load definition

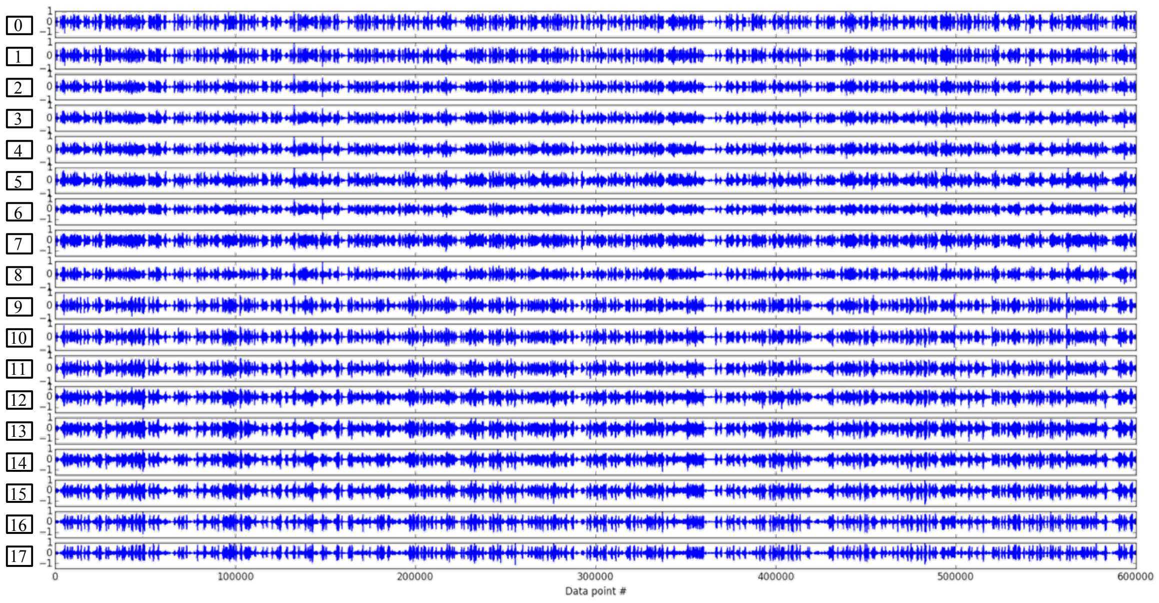


Figure 4. Simulated vertical acceleration measurements

Table 1. Random variables composing randomized traffic

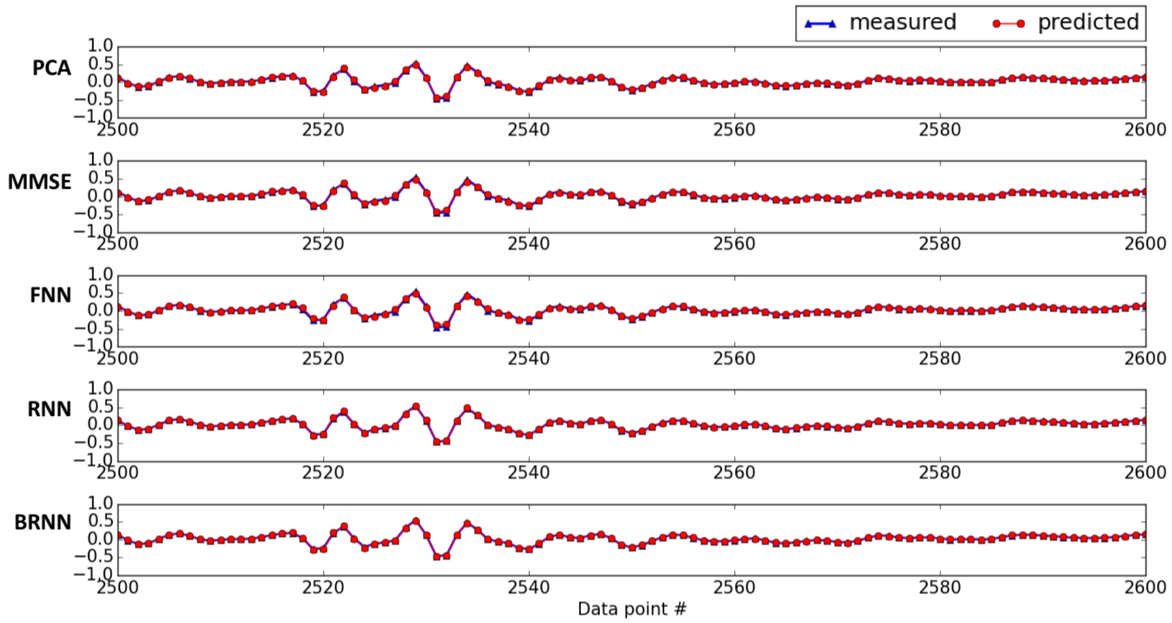
Factor	Values
Number of vehicles	1, 2, …, 9
Vehicle type	Auto, H-20, HS-20, HS-25
Vehicle speed	70 – 120 km/h (45 – 75 mph)
Vehicle lane	Three lanes (lane 1, lane 2 and lane 3)
Vehicle interval in a lane	1.5, 3.0, 4.5, 6.0 seconds

4.1 Evaluation of BRNN for sensor data reconstruction

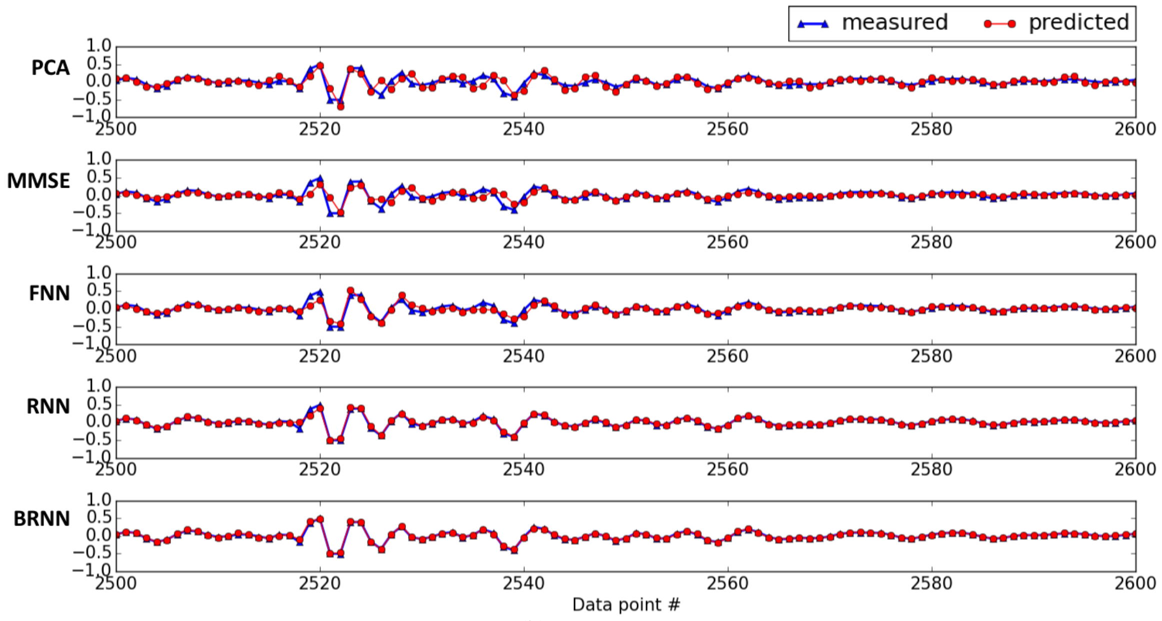
In this section, the BRNN-based sensor data reconstruction method is evaluated by comparing with other existing methods, including principal component analysis (PCA) [6], minimum mean square error (MMSE) estimation [7], feedforward neural network (FNN) [11] and recurrent neural network (RNN) [13]. Two BRNN models for reconstructing the data of Sensor 4 and Sensor 9 are created for the testing purpose. For comparison, PCA-, MMSE-, FNN- and RNN-based models corresponding to the two BRNN models are also created. The hyperparameters of neural network models, including FNN, RNN and BRNN, are determined heuristically, as listed in Table 2. Neural network models are constructed and trained using PyTorch [26].

Table 2. Hyperparameters of neural network models

	FNN	**RNN**	**BRNN**
Number of hidden layers	1	1	1
Number of hidden units	100	100	100 (50 forward and 50 backward hidden units)
Activation function for hidden layer	Hyperbolic tangent function	Hyperbolic tangent function	Hyperbolic tangent function
Activation function for output layer	Linear function	Linear function	Linear function
Optimization algorithm	Adam	Adam	Adam
Loss function	MSE	MSE	MSE
Learning rate	10^{-3}	10^{-3}	10^{-3}
Maximum number of epochs	200	200	200



(a) Sensor 4



(b) Sensor 9

Figure 5. Sensor data reconstruction results for different methods (PCA, MMSE, FNN, RNN and BRNN)

For each of the output target sensors (i.e., Sensors 4 and 9), five data reconstruction models (i.e., PCA, MMSE, FNN, RNN and BRNN) are trained with 80,000 data points (from data points 1 to 80,000) per sensor and tested with 40,000 data points (from data points 80,001 to 120,000) per sensor. Figure 5 shows the sensor data reconstruction results for the different methods. It should be noted that, for the sake of readability, the plots in Figure 5 show only 100 data points out of 40,000 reconstructed data points.

As can be seen in Figure 5(a), all methods work well for Sensor 4 which locates at the center of the bridge. For Sensor 9, which locates near the support, RNN and BRNN work better than the other methods as shown in Figure 5(b). Table 3 provides detailed information on the reconstruction results, including the computation time and the testing errors. A notable aspect shown in the results is that the testing error for the sensor at the center of the bridge (i.e., Sensor 4) is generally lower than the sensor located near the support of the bridge (i.e., Sensor 9). This is because the sensor located near the support has only one adjacent sensor, which means that there exists less spatial information for estimating the sensor data. Nevertheless, BRNN models yield much smaller RMSE error for Sensor 9 than all other methods, as the method takes both spatial and bidirectional temporal correlation into account. One tradeoff for the BRNN model is the amount of training time required. Once BRNN model is trained, the reconstruction process can be executed very efficiently even though the computing time remains higher than the other methods.

Table 3. Computing time and testing error of different sensor data reconstruction methods

	PCA	**MMSE**	**FNN**	**RNN**	**BRNN**
Training time per epoch (sec/epoch)	-	-	≈ 0.51	≈ 34.19	≈ 86.21
Total training time (sec)	≈ 0.053	≈ 0.026	≈ 510	≈ 6,838	≈ 17,242
Testing time (sec)	≈ 2.155	≈ 0.006	≈ 0.049	≈ 7.445	≈ 8.3971
RMSE for Sensor 4	0.0154	0.0151	0.0125	0.0034	0.0035
RMSE for Sensor 9	0.0658	0.0432	0.0308	0.0105	0.0070

4.2 Anomaly detection and isolation

To test the BRNN-based method for sensor validation problem, a faulty sensor scenario, in which the response datasets generated at Sensors 4 and 6 are corrupted with noise, is considered. Three test cases, each of which consists of 40,000 data points (selected from the data points from 480,001 to 600,000) per sensor are considered. Random noises are applied to each dataset of the test cases on Sensors 4 and 6 such that:

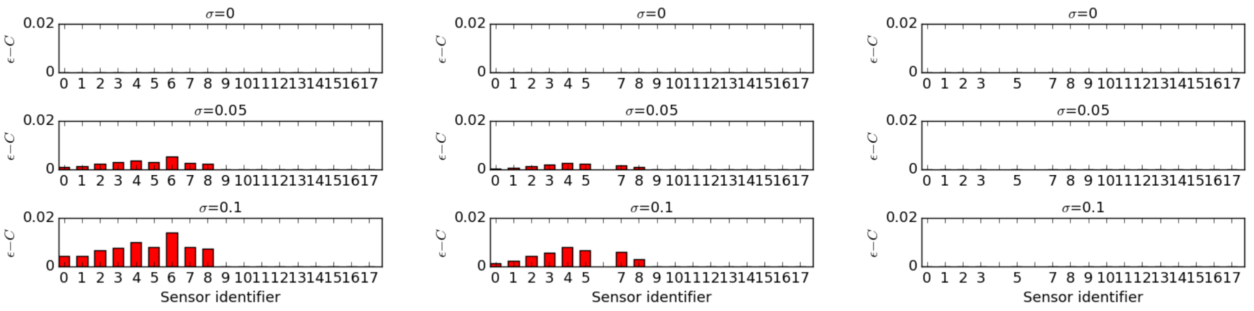
$$x'_i(t) \leftarrow x'_i(t) \times s, \qquad s \in N(1, \sigma^2) \tag{6}$$

where i is the index of the faulty sensors (i.e., 4 and 6) and $t = 480{,}001, \ldots, 600{,}000$. The level of noise is set by varying $\sigma = 0, 0.05, 0.1$. The experimental tests are conducted as follows:

(1) Baseline models with good quality data without noise are constructed. First, 18 BRNN models (i.e., 18 input-output combinations) are created by treating one sensor as the output while all other 17 sensors as input. For training, 80,000 data points (from the data points 1 to 80,000) per each sensor are employed.
(2) The confidence thresholds of the reconstruction errors are determined by computing the 95% confidence interval of the reconstruction errors for 10 test case scenarios, each has 40,000 data points (with good quality and without noise) selected from the data points 80,001 to 480,000.
(3) The reconstruction error for the three test cases with abnormal (noisy) datasets are computed for the 18 combinations of input and output sensor locations. For each test case, the computed reconstruction errors are then averaged over the entire dataset of 40,000 points. The reconstruction errors are compared with the thresholds.

Figure 6(a) shows the difference $(\epsilon - C)$ between the average reconstruction error ϵ and the threshold C at each sensor location. Here, the range of the y-axis is set to be greater than 0 to visualize only the reconstruction errors exceeding their thresholds. As shown in Figure 6(a), the reconstruction errors exceed the thresholds when the noise level $\sigma \geq 0.05$. As the noise level increases, the discrepancy between reconstruction error from the faulty data and the thresholds of the base model increases. It can also be seen that the discrepancies appear on all sensors (1-8) along the same girder with Sensors 4 and 6. This result is probably due to higher correlations among the response data on the same girder. This result also shows that while the anomaly is detected by comparing the reconstruction errors and thresholds, further analysis is needed to identify the location of faulty sensor.

To identify the faulty sensors, the dataset for Sensor 6, which has the highest discrepancy between the reconstruction errors from the noisy dataset and the threshold, is removed. For the remaining 17 sensors, the procedures for model training (with the 1 to 80,000 data points), computing the thresholds using 10 test cases each with 40,000 data points (from 80,001 to 480,000) and calculating the reconstruction errors with the three abnormal (noisy) datasets each with 40,000 data points (from 480,001 to 600,000) are repeated. Figure 6(b) shows the results without the dataset from Sensor 6. The results, however, still show detected anomalies, where Sensor 4 has the highest discrepancy between the reconstruction errors and its threshold. Therefore, the anomaly isolation procedure is continued by removing the dataset for Sensors 4 and 6. For the remaining 16 sensors, the procedures for model training, computing the threshold and calculating the reconstruction error are repeated with the same data points. Figure 6(c) shows the results without the datasets from Sensors 4 and 6. As no anomaly is detected after removing the dataset for Sensors 4 and 6, this implies that Sensors 4 and 6 are the potentially faulty sensors. This example shows that the data reconstruct using the BRNN method can detect and identify multiple faulty sensors.



(a) Anomaly detection with all sensors

(b) Anomaly detection after excluding Sensor 6

(c) Anomaly detection after excluding Sensor 4 and 6

Figure 6. Anomaly detection and isolation for the anomaly due to noise at Sensor 4 and Sensor 6

5. SUMMARY AND CONCLUSION

This paper discusses a data-driven sensor data reconstruction and anomaly detection method using bidirectional recurrent neural network. Unlike existing methods that typically use only the spatial correlation among the sensor data, the proposed method utilizes spatiotemporal correlation among sensor data to improve sensor data reconstruction accuracy. The bidirectional recurrent neural network capture spatiotemporal correlation in both positive time direction (i.e., past to present) and negative time direction (i.e., future to present). The BRNN-based sensor data reconstruction method consists of two phases, namely the training phase and the data reconstruction phase. In the training phase, a BRNN model is trained with training dataset that contains input sensors' data, as well as output sensor's data. In the data reconstruction phase, the trained BRNN model reads the input sensors' data and predicts the output sensor's data. The reconstructed sensor data can be used not only to recover faulty or missing sensor data, but also to detect and isolate anomalies by computing the difference between the measured sensor data and the reconstructed sensor data. The proposed method is demonstrated with the vibration data collected from numerical simulations that emulate a bridge structure under dynamic moving vehicle loads. The simulation results show that although with higher computational costs, the BRNN-based sensor data reconstruction method gives better accuracy on the predicted data than other existing data reconstruction methods, such as PCA-, MMSE-, FNN- and RNN-based methods. Furthermore, the results show that by combining with the analytical redundancy approach the sensor data reconstructed using BRNN-based method is able to detect and isolate anomalies for the faulty sensors.

ACKNOWLEDGEMENT

This research is supported by a Grant No. 13SCIPA01 from Smart Civil Infrastructure Research Program funded by Ministry of Land, Infrastructure and Transport (MOLIT) of Korea government and Korea Agency for Infrastructure Technology Advancement (KAIA). The research is also partially supported by a collaborative project funded by the US National Science Foundation (Grant No. ECCS-1446330 to Stanford University). The collaboration and support from the research team led by Prof. Jerome Lynch at University of Michigan are gratefully acknowledged. Any opinions, findings, conclusions or recommendations expressed in this paper are solely those of the authors and do not necessarily reflect the views of NSF, MOLIT, KAIA or any other organizations and collaborators.

REFERENCES

[1] V. Venkatasubramanian, R. Rengaswamy, S. Kavuri and K. Yin, "A review of process fault detection and diagnosis: Part III: Process history based methods," *Computers & chemical engineering,* vol. 27, no. 3, pp. 327-346, 2003.

[2] H. Sohn, C. Farrar, F. Hemez and J. Czarnecki, "A review of structural health review of structural health monitoring literature 1996-2001," Los Alamos National Laboratory, Los Alamos, NM, 2002.

[3] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Z. S. Nair, E. Kohler, G. Pottie, M. Hansen and M. Srivastava, "Sensor network data fault types," *ACM Transactions on Sensor Networks (TOSN),* vol. 5, no. 3, p. 25, 2009.

[4] S. Qin and W. Li, "Detection, identification, and reconstruction of faulty sensors with maximized sensitivity," *AIChE journal,* vol. 45, no. 9, pp. 1963-1976, 1999.

[5] A. Pouliezos and G. Stavrakakis, "Analytical redundancy methods," in *Real Time Fault Monitoring of Industrial Processes*, Dordrecht, Springer, 1994.

[6] G. Kerschen, P. De Boe, J. Golinval and K. Worden, "Sensor validation using principal component analysis," *Smart Materials and Structures,* vol. 14, no. 1, pp. 36-42, 2005.

[7] J. Kullaa, "Sensor validation using minimum mean square error estimation," *Mechanical Systems and Signal Processing,* vol. 24, no. 5, pp. 1444-1457, 2010.

[8] K. Law, S. Jeong and M. Ferguson, "A data-driven approach for sensor data reconstruction for bridge monitoring," in *2017 World Congress on Advances in Structural Engineering and Mechanics*, 2017.

[9] X. Xu, J. Hines and R. Uhrig, "Sensor validation and fault detection using neural networks," in *Maintenance and Reliability Conference (MARCON 99)*, 1999.

[10] I. Eski, S. Erkaya, S. Savas and S. Yildirim, "Fault detection on robot manipulators using artificial neural networks," *Robotics and Computer-Integrated Manufacturing,* vol. 27, no. 1, p. 115–123, 2011.

[11] K. Smarsly and K. H. Law, "Decentralized fault detection and isolation in wireless structural health monitoring systems using analytical redundancy," *Advances in Engineering Software,* vol. 73, p. 1–10, 2014.

[12] K. Dragos and K. Smarsly, "Distributed adaptive diagnosis of sensor faults using structural response data," *Smart Materials and Structures,* vol. 25, no. 10, p. 105019–15, 2016.

[13] A. I. Moustapha and R. R. Selmic, "Wireless Sensor Network Modeling Using Modified Recurrent Neural Networks: Application to Fault Detection," *IEEE Transactions on Instrumentation and Measurement,* vol. 57, no. 5, p. 981–988, 2008.

[14] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing,* vol. 45, no. 11, pp. 2673-2681, 1997.

[15] M. Hagan, H. Demuth, M. Beale and O. De Jess, Neural network design, 2014.

[16] Y. LeCun, L. Bottou, G. Orr and K. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, Berlin, Heidelberg., Springer, 1998, pp. 9-50.

[17] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE,* vol. 78, no. 10, pp. 1550-1560, 1990.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *The 3rd International Conference for Learning Representations*, 2015.

[19] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Transactions on Industrial Informatics,* vol. 9, no. 4, p. 2226–2238, 2013.

[20] A. B. Trunov and M. Polycarpou, "Automated fault diagnosis in nonlinear multivariable systems using a learning methodology," *IEEE Transactions on Neural Networks,* vol. 11, no. 1, p. 91–101, 2000.

[21] J. Kullaa, "Distinguishing between sensor fault, structural damage, and environmental or operational effects in structural health monitoring," *Mechanical Systems and Signal Processing,* vol. 25, no. 8, p. 2976–2989., 2011.

[22] Y. Zhang, S. O'Connor, G. van der Linden, A. Prakash and J. Lynch, "SenStore: a scalable cyberinfrastructure platform for implementation of data-to-decision frameworks for infrastructure health management," *Journal of Computing in Civil Engineering,* vol. 30, no. 5, p. p.04016012, 2016.

[23] S. O'Connor, Y. Zhang, J. Lynch, M. Ettouney and P. O. Jansson, "Long-term performance assessment of the Telegraph Road Bridge using a permanent wireless monitoring system and automated statistical process control analytics," *Structure and Infrastructure Engineering,,* vol. 13, no. 5, pp. 604-624., 2017.

[24] Computers and Structures, Inc., "Structural bridge design software | CSiBridge," [Online]. Available: https://www.csiamerica.com/products/csibridge. [Accessed 1 February 2018].

[25] Standard specifications for highway bridges, 17th edition ed., Washington D.C.: AASHTO, 2002.

[26] PyTorch, "PyTorch," [Online]. Available: http://pytorch.org/. [Accessed 1 February 2018].